

INFORMATICA

Clase de hoy

- Programación modular. Subprogramas:
 - Subrutinas
 - Funciones
- Módulos
- Makefile

Ejercicio previo

Caso 1

- Escribir un subprograma **subroutine** que, dado un vector x , calcule su media y desviación típica.

Caso 2

- Escribir un subprograma **function** que, dado un vector x , calcule su media.
- Escribir un subprograma **function** que, dado un vector x y su media, calcule su desviación típica.

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i, \quad dt = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2}$$

Programación modular

Tipos de unidades de programa.

- Programa principal: `program`
- Subprogramas:
 - Función: `function`
 - Subrutina: `subroutine`

Programación modular: Ejemplo

Escribid un programa que realice las siguientes tareas:

- b) Escribir un subprograma `function` que, dado un vector `x`, calcule su media.

```
function func_media(V,n)

    real*8 , intent(in)      :: V(n)
    integer, intent(in)      :: n

    real*8                    :: func_media

    func_media = sum(V)/n

end function
```

Programación modular: Ejemplo

Escribid un programa que realice las siguientes tareas:

- c) Escribir un subprograma `function` que, dado un vector `x` y su media, calcule su desviación típica.

```
function func_desviacion_1(V, n, med)

    real*8 , intent(in)      :: V(n)
    real*8 , intent(in)      :: med
    integer, intent(in)      :: n
    real*8                    :: func_desviacion_1

    integer :: i

    func_desviacion_1 = 0.d0
    do i = 1, N
        func_desviacion_1 = func_desviacion_1 + (V(i)-med)**2
    enddo
    func_desviacion_1 = sqrt(func_desviacion_1)

end function
```

Programación modular: Ejemplo

Escribid un programa que realice las siguientes tareas:

- c) Escribir un subprograma `function` que, dado un vector `x` y su media, calcule su desviación típica.

```
function func_desviacion_2(V)

    real*8 , intent(in)      :: V(:)
    real*8                  :: func_desviacion_2

    integer :: i,n
    real*8  :: valor_medio

    n = size(V,1)
    valor_medio = func_media(V,n)

    func_desviacion_2 = 0.d0
    do i = 1, N
        func_desviacion_2 = func_desviacion_2 + (V(i)-valor_medio)**2
    enddo
    func_desviacion_2 = sqrt(func_desviacion_2)

end function
```

Programación modular: Módulos

Un módulo es una unidad (normalmente se almacena en un fichero diferente) donde se encapsulan subprogramas:

```
module module_name
...
contains
    subprogram name#1
    ...
end subprogram name#1
    ...
    subprogram name#r
    ...
end subprogram name#r
end module module_name
```


Programación modular: Módulos

Para usar en el programa principal los subprogramas contenidos en el módulo hay que llamarlo antes de la declaración de las variables:

```
program main  
  
use module_name  
  
...  
  
end program main
```

Programación modular: Módulos

EXPLORADOR

EDITORES ABIERTOS 2 SIN...

ESTADISTICA

makefile

modulo.f95

principal.f95

principal.f95

```
1  program principal
2
3  use estadistica
4
5  real(8), allocatable
6  real(8)
7  integer
8
9  n = 10
10 pi = acos(-1.d0)
11 allocate(U(n))
12
13 do i = 1, N
14     U(i) = cos((i-1)*pi)
15 enddo
16
17 call estad(U,n, media,
18
19 write(*,*) 'Subrutina'
20 write(*,*) 'Media: ', m
21 write(*,*) 'Desviacion
22 write(*,*) '*****
23 write(*,*)
24
25 media      = func_medi
26 desv_tipica = func_desv
27
```

modulo.f95

```
1  module estadistica
2
3  contains
4
5  subroutine estad(V, n, med, desv)
6
7  real(8), intent(in) :: V(n)
8  integer, intent(in) :: N
9  real(8), intent(out):: med,desv
10
11
12 integer :: i
13
14 med = 0.d0
15 med = sum(V)/n
16
17 desv= 0.d0
18 do i = 1, N
19     desv = desv + (V(i)-med)**2
20 enddo
21 desv = sqrt(desv)
22
23 end subroutine
24
25 function func_media(V,n)
26
27     real(8), intent(in) :: V(n)
```

Programación modular: Módulos

EXPLORADOR

EDITORES ABIERTOS 2 SIN...

ESTADISTICA

makefile

modulo.f95

principal.f95

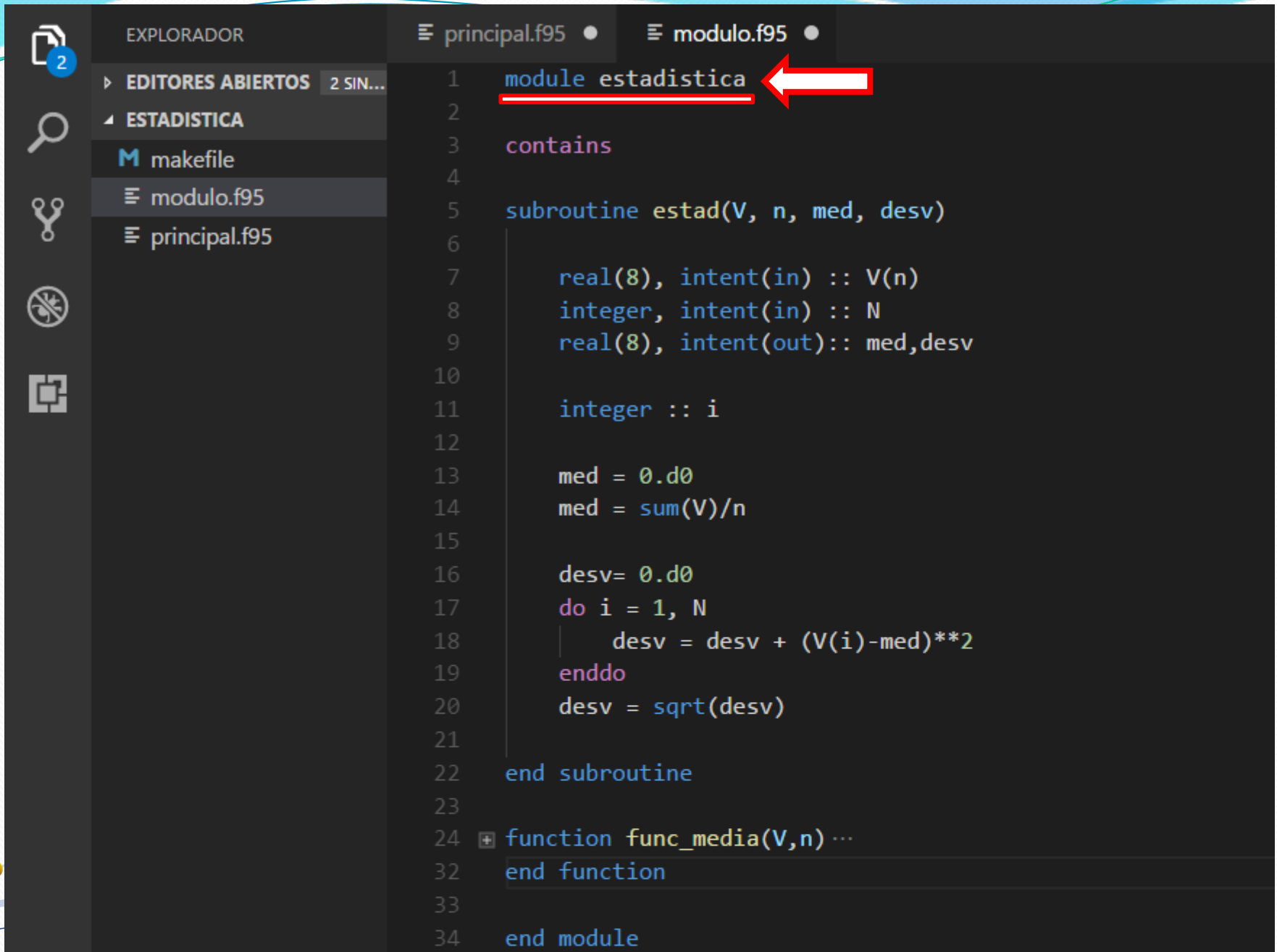
principal.f95

```
1  program principal
2
3  use estadistica
4
5  real(8), allocatable
6  real(8)
7  integer
8
9  n = 10
10 pi = acos(-1.d0)
11 allocate(U(n))
12
13 do i = 1, N
14     U(i) = cos((i-1)*pi)
15 enddo
16
17 call estad(U,n, media,
18
19 write(*,*) 'Subrutina'
20 write(*,*) 'Media: ', m
21 write(*,*) 'Desviacion
22 write(*,*) '*****
23 write(*,*)
24
25 media      = func_medi
26 desv_tipica = func_desv
27
```

modulo.f95

```
1  module estadistica
2
3  contains
4
5  subroutine estad(V, n, med, desv)
6
7  real(8), intent(in) :: V(n)
8  integer, intent(in) :: N
9  real(8), intent(out):: med,desv
10
11
12 integer :: i
13
14 med = 0.d0
15 med = sum(V)/n
16
17 desv= 0.d0
18 do i = 1, N
19     desv = desv + (V(i)-med)**2
20 enddo
21 desv = sqrt(desv)
22
23 end subroutine
24
25 function func_media(V,n)
26
27     real(8), intent(in) :: V(n)
```

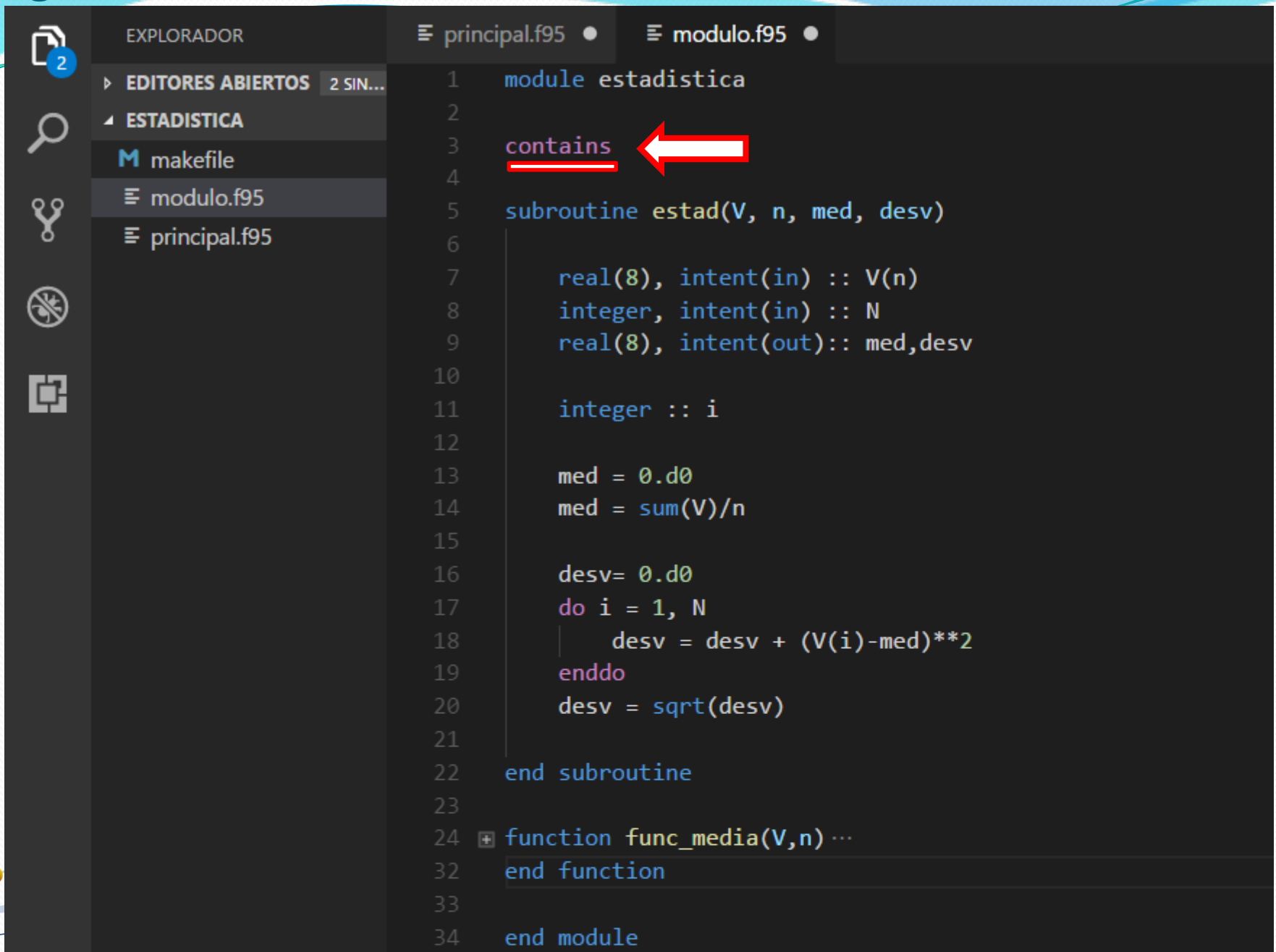
Programación modular: Módulos



The image shows a code editor interface with a dark theme. On the left, the 'EXPLORADOR' (Explorer) pane shows a project structure with 'EDITORES ABIERTOS' (2 SIN...) and 'ESTADISTICA' containing 'makefile', 'modulo.f95', and 'principal.f95'. The main editor window displays the content of 'modulo.f95'. A red arrow points to the line 'module estadistica' at the top of the code block.

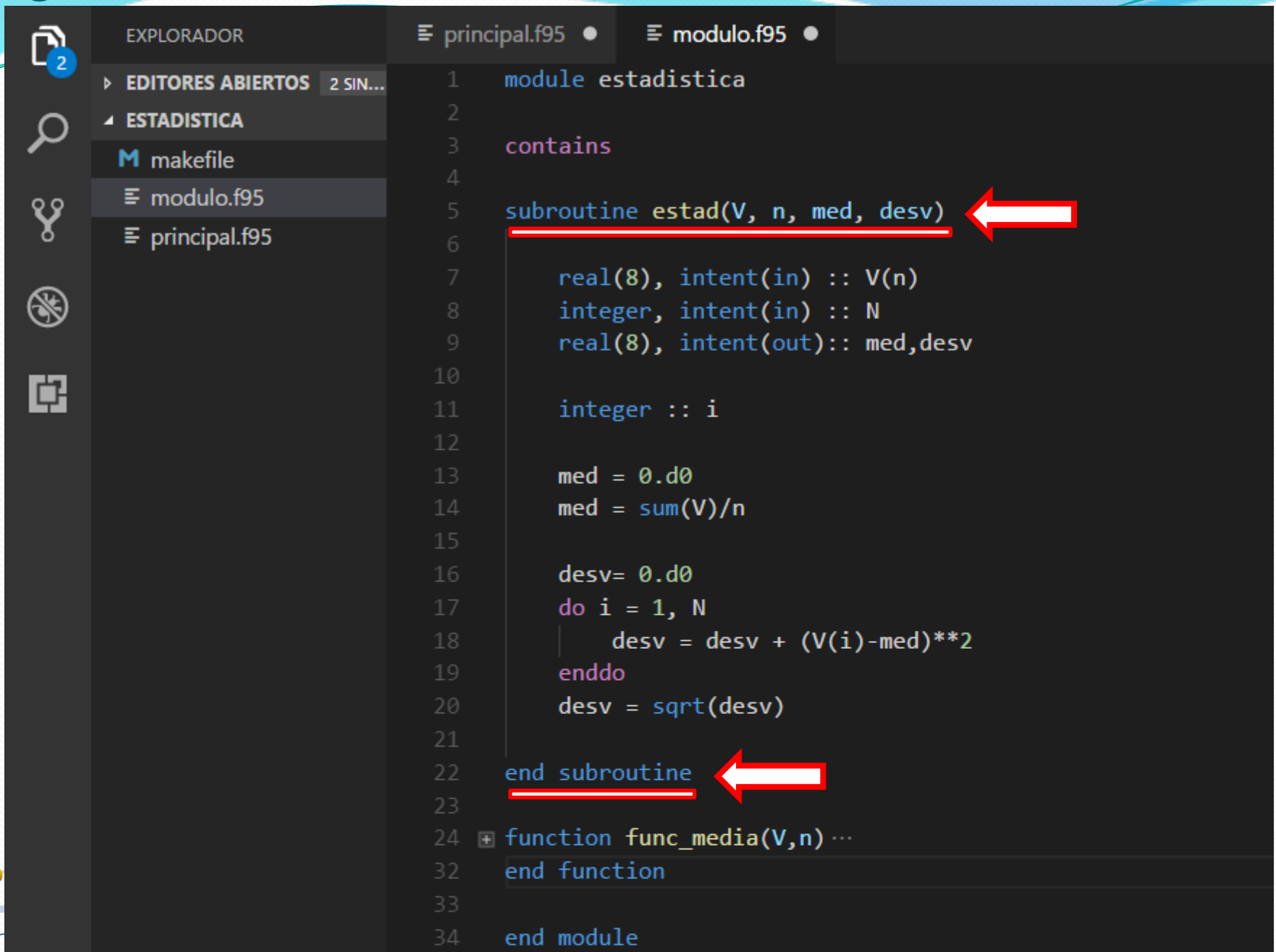
```
1  module estadistica
2
3  contains
4
5  subroutine estad(V, n, med, desv)
6
7      real(8), intent(in) :: V(n)
8      integer, intent(in) :: N
9      real(8), intent(out):: med, desv
10
11     integer :: i
12
13     med = 0.d0
14     med = sum(V)/n
15
16     desv= 0.d0
17     do i = 1, N
18         desv = desv + (V(i)-med)**2
19     enddo
20     desv = sqrt(desv)
21
22 end subroutine
23
24 + function func_media(V,n) ...
32 end function
33
34 end module
```

Programación modular: Módulos



```
1  module estadistica
2
3  contains
4
5  subroutine estad(V, n, med, desv)
6
7      real(8), intent(in) :: V(n)
8      integer, intent(in) :: N
9      real(8), intent(out):: med, desv
10
11      integer :: i
12
13      med = 0.d0
14      med = sum(V)/n
15
16      desv= 0.d0
17      do i = 1, N
18          desv = desv + (V(i)-med)**2
19      enddo
20      desv = sqrt(desv)
21
22  end subroutine
23
24  + function func_media(V,n) ...
32  end function
33
34  end module
```

Programación modular: Módulos

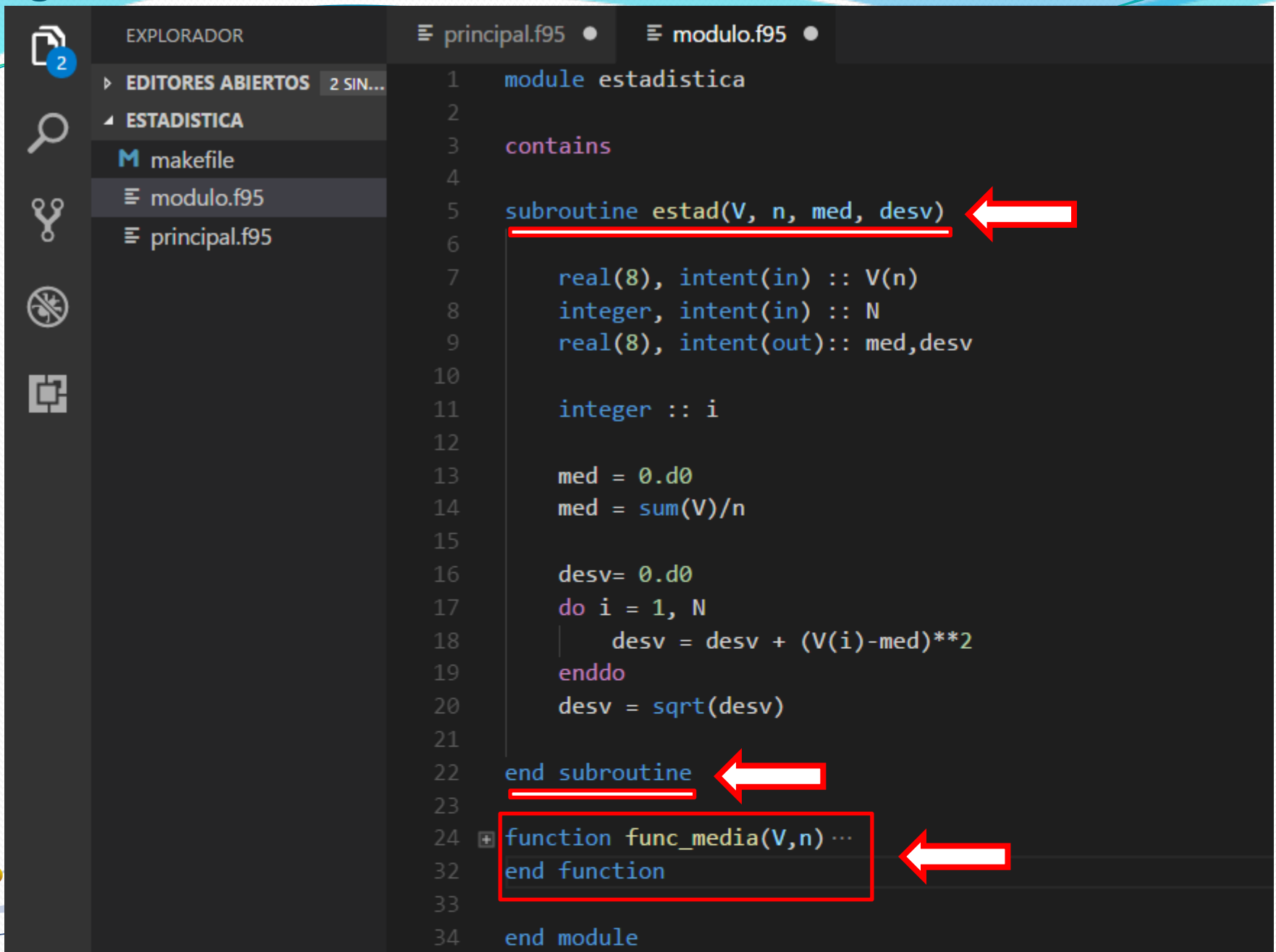


The screenshot shows an IDE with a dark theme. On the left is a sidebar with icons for Explorer, Search, Source Control, Run and Debug, and Extensions. The Explorer panel shows a project named 'ESTADISTICA' with files 'makefile', 'modulo.f95', and 'principal.f95'. The 'modulo.f95' file is open in the editor. The code in the editor is as follows:

```
1  module estadistica
2
3  contains
4
5  subroutine estad(V, n, med, desv)
6
7      real(8), intent(in) :: V(n)
8      integer, intent(in) :: N
9      real(8), intent(out):: med, desv
10
11      integer :: i
12
13      med = 0.d0
14      med = sum(V)/n
15
16      desv= 0.d0
17      do i = 1, N
18          desv = desv + (V(i)-med)**2
19      enddo
20      desv = sqrt(desv)
21
22  end subroutine
23
24  function func_media(V,n) ...
25
26  end function
27
28
29
30
31
32
33
34  end module
```

Two red arrows are present: one pointing to the line `subroutine estad(V, n, med, desv)` and another pointing to the line `end subroutine`.

Programación modular: Módulos



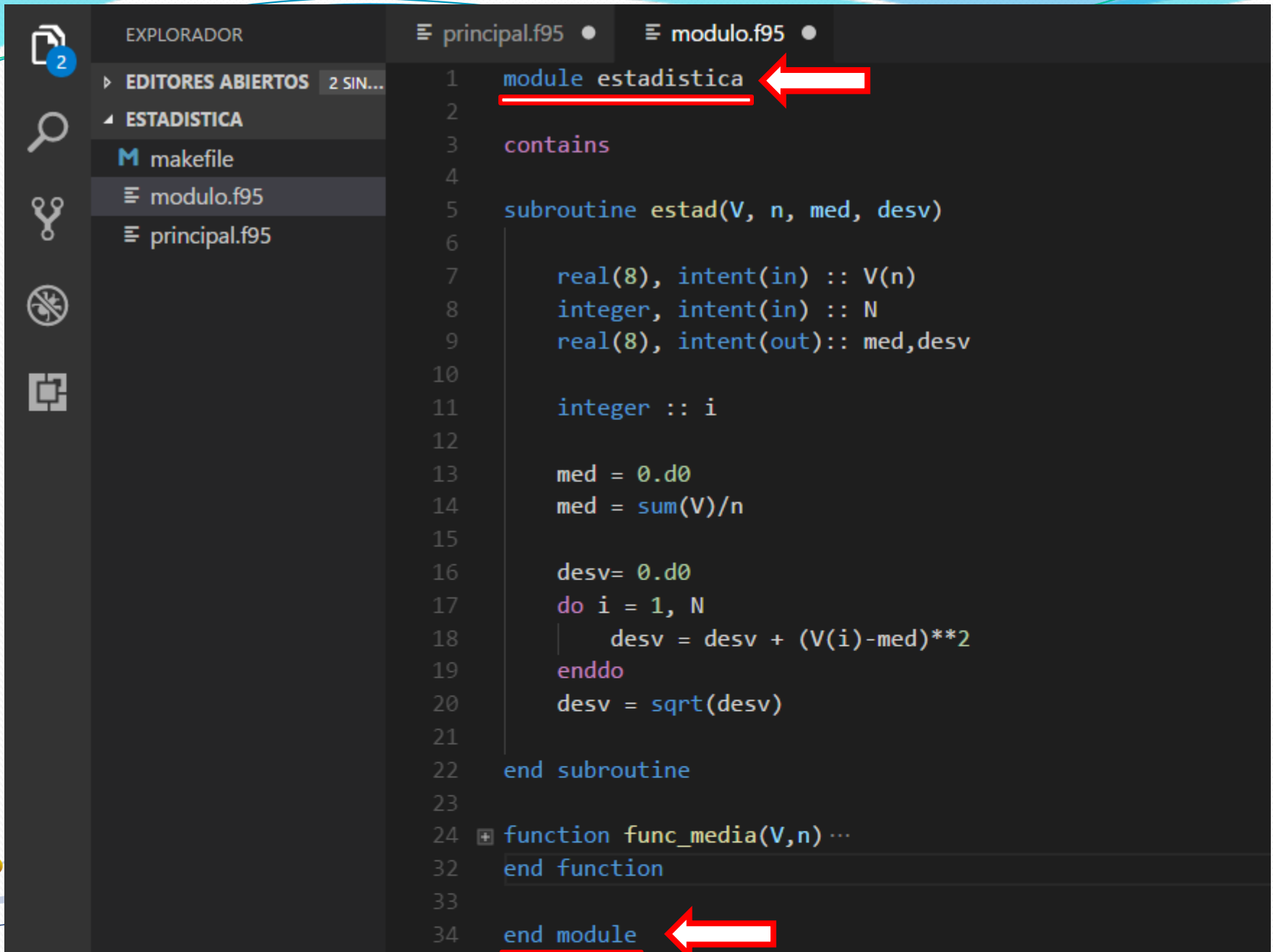
The screenshot shows a code editor with two tabs: 'principal.f95' and 'modulo.f95'. The 'modulo.f95' tab is active, displaying the following Fortran code:

```
1  module estadistica
2
3  contains
4
5  subroutine estad(V, n, med, desv)
6
7      real(8), intent(in) :: V(n)
8      integer, intent(in) :: N
9      real(8), intent(out):: med, desv
10
11      integer :: i
12
13      med = 0.d0
14      med = sum(V)/n
15
16      desv= 0.d0
17      do i = 1, N
18          desv = desv + (V(i)-med)**2
19      enddo
20      desv = sqrt(desv)
21
22  end subroutine
23
24  function func_media(V,n) ...
25
26  end function
27
28  end module
```

Red arrows highlight the following parts of the code:

- Line 5: `subroutine estad(V, n, med, desv)`
- Line 22: `end subroutine`
- Line 24: `function func_media(V,n) ...`

Programación modular: Módulos



The screenshot shows an IDE with a dark theme. On the left, the 'EXPLORADOR' (Explorer) pane shows a project named 'ESTADISTICA' containing files 'makefile', 'modulo.f95', and 'principal.f95'. The 'modulo.f95' file is selected. The main editor pane shows the code for 'modulo.f95'. The code defines a module named 'estadistica' which contains a subroutine 'estad' and a function 'func_media'. The subroutine 'estad' calculates the mean and standard deviation of an array 'V' of size 'n'. The function 'func_media' is partially visible at the bottom. Two red arrows point to the 'module estadistica' line at the top and the 'end module' line at the bottom, highlighting the module boundaries.

```
1  module estadistica
2
3  contains
4
5  subroutine estad(V, n, med, desv)
6
7      real(8), intent(in) :: V(n)
8      integer, intent(in) :: N
9      real(8), intent(out):: med, desv
10
11     integer :: i
12
13     med = 0.d0
14     med = sum(V)/n
15
16     desv= 0.d0
17     do i = 1, N
18         desv = desv + (V(i)-med)**2
19     enddo
20     desv = sqrt(desv)
21
22 end subroutine
23
24 + function func_media(V,n) ...
32 end function
33
34 end module
```


Programación modular: Módulos

EXPLORADOR

EDITORES ABIERTOS 2 SIN...

ESTADISTICA

makefile

modulo.f95

principal.f95

principal.f95

```
1  program principal
2
3  use estadistica
4
5  real(8), allocatable
6  real(8)
7  integer
8
9  n = 10
10 pi = acos(-1.d0)
11 allocate(U(n))
12
13 do i = 1, N
14     U(i) = cos((i-1)*pi)
15 enddo
16
17 call estad(U,n, media,
18
19 write(*,*) 'Subrutina'
20 write(*,*) 'Media: ', m
21 write(*,*) 'Desviacion
22 write(*,*) '*****
23 write(*,*)
24
25 media      = func_medi
26 desv_tipica = func_desv
27
```

modulo.f95

```
1  module estadistica
2
3  contains
4
5  subroutine estad(V, n, med, desv)
6
7  real(8), intent(in) :: V(n)
8  integer, intent(in) :: N
9  real(8), intent(out):: med,desv
10
11
12 integer :: i
13
14 med = 0.d0
15 med = sum(V)/n
16
17 desv= 0.d0
18 do i = 1, N
19     desv = desv + (V(i)-med)**2
20 enddo
21 desv = sqrt(desv)
22
23 end subroutine
24
25 function func_media(V,n)
26
27     real(8), intent(in) :: V(n)
```

Programación modular: Módulos

EXPLORADOR

EDITORES ABIERTOS 2 SIN...

ESTADISTICA

makefile

modulo.f95

principal.f95

principal.f95

```
1  program principal
2
3  use estadistica
4
5  real(8), allocatable
6  real(8)
7  integer
8
9  n = 10
10 pi = acos(-1.d0)
11 allocate(U(n))
12
13 do i = 1, N
14     U(i) = cos((i-1)*pi)
15 enddo
16
17 call estad(U,n, media,
18
19 write(*,*) 'Subrutina'
20 write(*,*) 'Media: ', m
21 write(*,*) 'Desviacion
22 write(*,*) '*****
23 write(*,*)
24
25 media      = func_medi
26 desv_tipica = func_desv
27
```

modulo.f95

```
1  module estadistica
2
3  contains
4
5  subroutine estad(V, n, med, desv)
6
7  real(8), intent(in) :: V(n)
8  integer, intent(in) :: N
9  real(8), intent(out):: med,desv
10
11
12 integer :: i
13
14 med = 0.d0
15 med = sum(V)/n
16
17 desv= 0.d0
18 do i = 1, N
19     desv = desv + (V(i)-med)**2
20 enddo
21 desv = sqrt(desv)
22
23 end subroutine
24
25 function func_media(V,n)
26
27     real(8), intent(in) :: V(n)
```

Programación modular: Módulos

principal.f95 x

```
1  program principal
2
3  use estadistica
4
5  real(8), allocatable :: U(:)
6  real(8) :: media, desv_tipica, pi
7  integer :: i,n
8
9  n = 8
10 pi = acos(-1.d0)
11 allocate(U(n))
12
13 do i = 1, N
14     U(i) = cos((i-1)*pi/N)
15 enddo
16
17 call estad(U,n, media, desv_tipica)
18
19 write(*,*) 'Subrutina'
20 write(*,*) 'Media: ', media
21 write(*,*) 'Desviacion tipica: ', desv_tipica
22 write(*,*) '*****'
23 write(*,*)
24
25 media = func_media(U,n)
26 desv_tipica = func_desviacion_1(U,n,media)
27
28 write(*,*) 'Funciones'
```

modulo.f95 x

```
1  module estadistica
2
3  contains
4
5  subroutine estad(V, n, med, desv)
6
7      real(8), intent(in) :: V(n)
8      integer, intent(in) :: N
9      real(8), intent(out):: med,desv
10
11      integer :: i
12
13      med = 0.d0
14      med = sum(V)/n
15
16      desv= 0.d0
17      do i = 1, N
18          desv = desv + (V(i)-med)**2
19      enddo
20      desv = sqrt(desv)
21
22  end subroutine
23
24  function func_media(V,n) ...
25  end function
26
27  function func_desviacion_1(V,n,med)
28
29  end function
```

Programación modular: Módulos

```
principal.f95 •  
  
program principal  
  
  use estadistica  
  use variables  
  
  n = 8  
  pi = acos(-1.d0)  
  allocate(U(n))  
  
  do i = 1, N  
    U(i) = cos((i-1)*pi/N)  
  enddo  
  
  call estad(U,n, media, desv_tipica)  
  
  write(*,*) 'Subrutina'  
  write(*,*) 'Media: ', media  
  write(*,*) 'Desviacion tipica: ', desv_tipica  
  write(*,*) '*****'  
  write(*,*)
```

```
modulo2.f95 •  
  
1  module variables  
2  
3  real(8), allocatable :: U(:)  
4  real(8) :: media, desv_tipica, pi  
5  integer :: i,n  
6  
7  end module
```

```
modulo.f95 x  
  
1  module estadistica  
2  
3  contains  
4  
5  subroutine estad(V, n, med, desv)  
6  
7      real(8), intent(in) :: V(n)  
8      integer, intent(in) :: N  
9      real(8), intent(out):: med,desv  
10  
11      i
```

Un módulo también puede contener variables que serán visibles en todas las unidades de programa que incluyan ese módulo

Programación modular: Módulos

```
principal.f95 •
program principal

use estadistica
use variables ←

n = 8
pi = acos(-1.d0)
allocate(U(n))

do i = 1, N
    U(i) = cos((i-1)*pi/N)
enddo

call estad(U,n, media, desv_tipica)

write(*,*) 'Subrutina'
write(*,*) 'Media: ', media
write(*,*) 'Desviacion tipica: ', desv_tipica
write(*,*) '*****'
write(*,*)
```

```
modulo2.f95 •
1  module variables ←
2
3  real(8), allocatable :: U(:)
4  real(8)               :: media, desv_tipica, pi
5  integer               :: i,n
6
7  end module
```

```
modulo.f95 x
1  module estadistica
2
3  contains
4
5  subroutine estad(V, n, med, desv)
6
7      real(8), intent(in) :: V(n)
8      integer, intent(in) :: N
9      real(8), intent(out):: med,desv
10
11      i
```

Un módulo también puede contener variables que serán visibles en todas las unidades de programa que incluyan ese módulo

Programación modular: Módulos

```
principal.f95 •  
  
program principal  
  
use estadistica ←  
use variables  
  
n = 8  
pi = acos(-1.d0)  
allocate(U(n))  
  
do i = 1, N  
    U(i) = cos((i-1)*pi/N)  
enddo  
  
call estad(U,n, media, desv_tipica)  
  
write(*,*) 'Subrutina'  
write(*,*) 'Media: ', media  
write(*,*) 'Desviacion tipica: ', desv_tipica  
write(*,*) '*****'  
write(*,*)
```

```
modulo2.f95 •  
  
1 module variables  
2  
3 real(8), allocatable :: U(:)  
4 real(8) :: media, desv_tipica, pi  
5 integer :: i,n  
6  
7 end module
```

```
modulo.f95 x  
  
1 module estadistica ←  
2  
3 contains  
4  
5 subroutine estad(V, n, med, desv)  
6  
7     real(8), intent(in) :: V(n)  
8     integer, intent(in) :: N  
9     real(8), intent(out):: med,desv  
10  
11     i
```

Un módulo también puede contener variables que serán visibles en todas las unidades de programa que incluyan ese módulo

Programación modular:

2.- Compilar y construir. *Makefile*

```
# FORTRAN compiler given below
FC = gfortran
# LINKER GIVEN BELOW
LD = gfortran
# COMPILER FLAGS GIVEN BELOW
FFLAGS = -fbounds-check
# COMMAND TO DELETE
RM = del

SRC = modulo.f95 principal.f95

OBJ = $(SRC:.f95=.o)

EXE = executable

.SUFFIXES: .f95 .o

all: $(OBJ) main

%.o : %.f95
    $(FC) $(FFLAGS) -c $<

main : $(OBJ)
    $(LD) $(FFLAGS) $(OBJ) -o $(EXE)

clean:    $(RM) -f $(OBJ) *.mod
```

Programación modular:

2.- Compilar y construir. *Makefile*

makefile

```
# FORTRAN compiler given below
FC = gfortran
# LINKER GIVEN BELOW
LD = gfortran
# COMPILER FLAGS GIVEN BELOW
FFLAGS = -fbounds-check
# COMMAND TO DELETE
RM = del

SRC = modulo.f95 principal.f95

OBJ = $(SRC:.f95=.o)

EXE = executable

.SUFFIXES: .f95 .o

all: $(OBJ) main

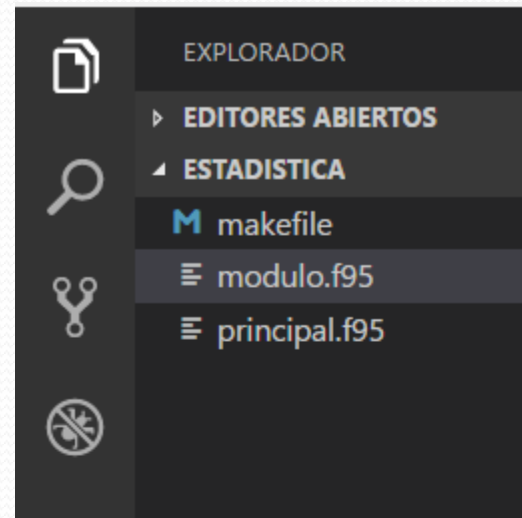
%.o : %.f95
    $(FC) $(FFLAGS) -c $<

main : $(OBJ)
    $(LD) $(FFLAGS) $(OBJ) -o $(EXE)

clean:
    $(RM) -f $(OBJ) *.mod
```

Usuarios > uss > Javier > Trabajo > Informatica17-18 > Codigos > estadistica

Nombre	Fecha de modifica...	Tipo
makefile	22/11/2017 11:55	Archivo
modulo	22/11/2017 11:53	Archivo F95
principal	22/11/2017 11:52	Archivo F95



Programación modular: makefile

Makefile — Ejemplo

EXPLORER

OPEN EDITORS

Welcome

principal.f95

Makefile

EJEMPLO

estadistica.mod

executable

Makefile

modulo.f95

modulo.o

principal.f95

principal.o

Welcome

principal.f95

Makefile

```
1  # FORTRAN compiler given below
2  FC = gfortran
3
4  # LINKER GIVEN BELOW
5  LD = gfortran
6
7  # COMPILER FLAGS GIVEN BELOW
8  FFLAGS = -fbounds-check
9
10 # COMMAND TO DELETE
11 RM = rm
12
13 SRC = modulo.f95 principal.f95
14
15 OBJ = $(SRC:.f95=.o)
16 EXE = executable
17
18 .SUFFIXES: .f95 .o
19
20 all: $(OBJ) main
21
22 %.o : %.f95
23 |     $(FC) $(FFLAGS) -c $<
24
25 main : $(OBJ)
26 |     $(LD) $(FFLAGS) $(OBJ) -o $(EXE)
27
28 clean:
29 |     $(RM) -f $(OBJ) *.mod
30
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

```
grubio@anoat:~/UPMdrive/Teaching/Lecturer/Curso1718/Informatica/FirstSemester/Clase_10/Ejemplo$ make all
gfortran -fbounds-check -c modulo.f95
gfortran -fbounds-check -c principal.f95
gfortran -fbounds-check modulo.o principal.o -o executable
grubio@anoat:~/UPMdrive/Teaching/Lecturer/Curso1718/Informatica/FirstSemester/Clase_10/Ejemplo$
```

Programación modular: makefile

The screenshot shows the Visual Studio Code interface with a Makefile open. The Explorer sidebar on the left shows the project structure, with 'executable' highlighted under the 'EJEMPLO' folder. A red arrow points from 'executable' to the 'EXEC = executable' line in the Makefile. Another red box highlights 'RM = rm' in the Makefile. A text overlay states 'Windows: del // MacOS: rm'. A large green arrow points down to the terminal at the bottom, which shows the command 'make all' being executed successfully.

Makefile — Ejemplo

EXPLORER

OPEN EDITORS

- Welcome
- principal.f95
- Makefile

EJEMPLO

- estadistica.mod
- executable
- Makefile
- modulo.f95
- modulo.o
- principal.f95
- principal.o

```
1 # FORTRAN compiler given below
2 FC = gfortran
3
4 # LINKER GIVEN BELOW
5 LD = gfortran
6
7 # COMPILER FLAGS GIVEN BELOW
8 FFLAGS = -fbounds-check
9
10 # COMMAND TO DELETE
11 RM = rm
12
13 SRC = modulo.f95 principal.f95
14
15 OBJ = $(SRC:.f95=.o)
16 EXE = executable
17
18 .SUFFIXES: .f95 .o
19
20 all: $(OBJ) main
21
22 %.o : %.f95
23     $(FC) $(FFLAGS) -c $<
24
25 main : $(OBJ)
26     $(LD) $(FFLAGS) $(OBJ) -o $(EXE)
27
28 clean:
29     $(RM) -f $(OBJ) *.mod
30
```

Windows: del // MacOS: rm

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
grubio@anoat:~/UPMdrive/Teaching/Lecturer/Curso1718/Informatica/FirstSemester/Clase_10/Ejemplo$ make all
gfortran -fbounds-check -c modulo.f95
gfortran -fbounds-check -c principal.f95
gfortran -fbounds-check modulo.o principal.o -o executable
grubio@anoat:~/UPMdrive/Teaching/Lecturer/Curso1718/Informatica/FirstSemester/Clase_10/Ejemplo$
```

Programación modular. Trabajo para casa

Escribir una subrutina que calcule el producto de dos matrices cualesquiera. Podéis utilizar *matmul* dentro de la subrutina.

La subrutina debe comprobar que las dimensiones de las matrices permiten multiplicarlas. La cabecera de la subrutina será:

```
subroutine producto (A, B, AxB, info )  
    real , intent (in) :: A(: , :)  
    real , intent (in) :: B(: , :)  
    real , intent (out) :: AxB (: , :)  
    integer, intent(out):: info  
end subroutine producto
```

Info contiene información sobre el resultado de la multiplicación: se ha podido llevar a cabo, las dimensiones de A y B no son las adecuadas... El significado de los distintos valores deberá estar comentado en el código.

Escribir otra subrutina que, mediante llamadas a la anterior, halle la potencia p-ésima de una matriz cuadrada.

Programación modular. Trabajo para casa

- Cread un modulo llamado algebra_lineal donde guardaréis las subrutinas de multiplicación de matrices y potencia.
- Cread un programa principal que compruebe el siguiente resultado:

$$A = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

$$A^k = \begin{pmatrix} 2^{k-1} & 0 & 2^{k-1} \\ 0 & 1 & 0 \\ 2^{k-1} & 0 & 2^{k-1} \end{pmatrix} \quad k \in \mathbb{N}$$

- Cread un makefile adecuado y ejecutad el código.